Fast Byzantine Leader Election in Dynamic Networks

Fast Byzantine Leader Election in Dynamic Networks

John Augustine[1,⋆], Gopal Pandurangan[2,⋆⋆], and Peter Robinson[3,⋆⋆⋆]

[1] Indian Inst. of Tech. Madras, Chennai, TN, India
[2] Department of Computer Science, University of Houston, TX, USA
[3] Queen's University Belfast, United Kingdom

Abstract. We study the fundamental Byzantine leader election problem in dynamic networks where the topology can change from round to round and nodes can also experience heavy *churn* (i.e., nodes can join and leave the network continuously over time). We assume the full information model where the Byzantine nodes have complete knowledge about the entire state of the network at every round (including random choices made by all the nodes), have unbounded computational power and can deviate arbitrarily from the protocol. The churn is controlled by an adversary that has complete knowledge and control over which nodes join and leave and at what times and also may rewire the topology in every round and has unlimited computational power, but is oblivious to the random choices made by the algorithm.

Our main contribution is an $O(\log^3 n)$ round algorithm that achieves Byzantine leader election under the presence of up to $O(n^{1/2-\varepsilon})$ Byzantine nodes (for a small constant $\varepsilon > 0$) and a churn of up to $O(\sqrt{n}/\operatorname{polylog}(n))$ nodes per round (where n is the stable network size). The algorithm elects a leader with probability at least $1 - n^{-\Omega(1)}$ and guarantees that it is an honest node with probability at least $1 - n^{-\Omega(1)}$; assuming the algorithm succeeds, the leader's identity will be known to a $1 - o(1)$ fraction of the honest nodes. Our algorithm is fully-distributed, lightweight, and is simple to implement. It is also scalable, as it runs in polylogarithmic (in n) time and requires nodes to send and receive messages of only polylogarithmic size per round. To the best of our knowledge, our algorithm is the first scalable solution for Byzantine leader election in a dynamic network with a high rate of churn; our protocol can also be used to solve Byzantine agreement in a straightforward way. We also show how to implement an (almost-everywhere) public coin with constant bias in a dynamic network with Byzantine nodes and provide a mechanism for enabling honest nodes to store information reliably in the network, which might be of independent interest.

⋆ John Augustine was supported by IIT Madras New Faculty Seed Grant, IIT Madras Exploratory Research Project, and Indo-German Max Planck Center for Computer Science (IMPECS).

⋆⋆ Gopal Pandurangan was supported in part by NSF grant CCF-1527867.

⋆⋆⋆ Peter Robinson was partly supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under the ASAP project, grant agreement no. 619706.

1 Introduction

Motivated by the need for robust and secure distributed computation in large-scale (sparse) networks such as peer-to-peer (P2P) and overlay networks, we study the fundamental *Byzantine leader election* problem in *dynamic* networks, where a large number of nodes can join and leave the network continuously and the topology can also change continuously. The Byzantine leader election problem in dynamic networks is challenging because the goal is to guarantee that an honest (i.e., non-Byzantine) node is elected as a leader with probability at least $1-o(1)$ and whose identity is known to *most* honest nodes[4] despite the adversarial network dynamism and the presence of Byzantine nodes. Byzantine leader election is related to another fundamental and central problem in distributed computing, namely, *Byzantine agreement*. In fact, in our setting, Byzantine leader election is a harder problem, since it can be used to solve *almost-everywhere* Byzantine agreement in a straightforward way.

Byzantine agreement and leader election have been challenging problems even in static networks. Indeed, until recently, almost all the work known in the literature (see e.g., [14, 19, 20, 22, 31]) have addressed the Byzantine almost-everywhere agreement problem only in static networks. Unfortunately, these approaches fail in dynamic networks where both nodes *and* edges can change by a large amount in *every* round. For example, Upfal [31] showed how one can achieve almost-everywhere agreement under up to a linear number — up to εn, for a sufficiently small $\varepsilon > 0$ — of Byzantine faults in a bounded-degree expander network (n is the network size). However, the algorithm requires knowledge of the global topology, since at the start, nodes need to have this information hardcoded. The work of King et al. [23] is important in the context of P2P networks, as it was the first to study scalable (polylogarithmic communication and number of rounds) algorithms for Byzantine leader election and agreement. However, as pointed out by the authors, their algorithm works only for static networks. Similar to Upfal's algorithm, the nodes require hardcoded information on the network topology to begin with and thus the algorithm does not work when the topology changes (in particular, when the edges are also changing in every round). In fact, this work ([23]) raised the open question of whether one can design Byzantine leader election and agreement protocols that can work in highly dynamic networks with a large churn rate.

The work of [4] was the first to study the Byzantine agreement problem in a dynamic network with a large churn rate. However, this algorithm does not directly solve the *leader election* problem, since the value that (most of) the honest nodes agree may be a value that was generated by a Byzantine node; using the agreement algorithm in a straightforward way does not give any guarantee that an honest node will be elected as leader. Hence, a more involved approach is needed for Byzantine leader election.

[4] In sparse, bounded-degree networks, an adversary can always isolate some number of honest nodes, hence "almost-everywhere" is the best one can hope for in such networks (cf. [14]).

Our Main Result. We study Byzantine leader election in dynamic networks where the topology can change from round to round and nodes can also experience heavy *churn* (i.e., nodes can join and leave the network continuously over time). Our goal is to design a fast distributed algorithm (running in a small number of rounds) that guarantees, despite a relatively large number of Byzantine nodes and high node churn, that an honest node is elected as leader and almost all honest nodes know the identity of this leader.

Before we state our results, we briefly describe the key ingredients of our model here. (Our model is described in detail in Section 1.1, it is similar to the model considered in prior work, e.g., [6, 4, 5].) We consider a dynamic network as a sparse bounded degree *expander* graph whose topology — *both nodes and edges* — can change arbitrarily from round to round and is controlled by an adversary. However, we assume that the total number of nodes in the network is stable. Note that our model is quite general in the sense that we only assume that the topology is an expander[5] at every step; no other special properties are assumed. Indeed, expanders have been used extensively to model dynamic P2P networks in which the expander property is preserved under insertions and deletions of nodes (e.g., [26, 29]). Since we do not make assumptions on how the topology is preserved, our model is applicable to all such expander-based networks. (We note that various prior work on dynamic network models make similar assumptions on preservation of topological properties — such as connectivity, expansion etc. — at every step under dynamic *edge* insertions/deletions — cf. Section 1. The issue of how such properties are preserved are abstracted away from the model, which allows one to focus on the dynamism. Indeed, this abstraction has been a feature of most dynamic models e.g., see the survey of [10].) Furthermore, our results are applicable to dynamic network models with good expansion where only edges change (and no churn) —such models have been studied extensively in recent years (cf. Section 1).

The number of node changes *per round* is called the *churn rate* or *churn limit*. We consider a churn rate of up to $O(\sqrt{n}/\text{polylog}(n))$, where n is the stable network size. Furthermore, we assume that a large number of nodes can be *Byzantine*. We allow up to $O(n^{\frac{1}{2}-\varepsilon})$ Byzantine nodes in any round, where $\varepsilon > 0$ is a small constant. Byzantine nodes (who have unbounded computational power) are "adaptive", in the sense that they know the entire states of all nodes at the beginning of every round and thus can take the current state of the network into account when determining their next action. In each round, an *oblivious adversary* chooses some $O(\sqrt{n}/\text{polylog}(n))$ nodes that are replaced by new nodes. The oblivious adversary has complete control over what nodes join and leave and at what time and also may rewire the edges in every round and has unlimited computational power but is oblivious to the random choices of the nodes. (Note that an adaptive churn adversary that knows the current state of all the nodes is not very interesting in the context of leader election, since it can churn out the leader as soon as it is elected.)

[5] In principle, our results can potentially be extended to graphs with weaker expansion guarantees as well; however the amount of churn and Byzantine nodes that can be tolerated will be reduced correspondingly.

Our main contribution is a randomized distributed algorithm that achieves leader election with high probability[6] even under a large number of Byzantine nodes and continuous adversarial churn in a number of rounds that is polylogarithmic in n (where n is the stable network size). In particular, we show the following theorem:

Theorem 1 (Main Theorem). *Let $\varepsilon > 0$ be any fixed constant. Consider a synchronous dynamic n-node (n is the stable network size) expander network where up to $O(n^{\frac{1}{2}-\varepsilon})$ nodes are Byzantine (who can behave arbitrarily and who have full knowledge of the current network state including past random choices made by other nodes), and suppose that up to $O(\sqrt{n}/\operatorname{polylog}(n))$ nodes are subjected to churn per round determined by an oblivious adversary. There exists an algorithm that elects a leader with probability $1 - n^{-\Omega(1)}$ who is known by all except $o(n)$ nodes, and the leader is an honest node with probability $1 - n^{-\Omega(1)}$. The algorithm runs in $O(\log^3 n)$ rounds and uses messages of $O(\operatorname{polylog}(n))$ size.*

Our algorithm is the first-known, decentralized Byzantine leader election algorithm in highly dynamic networks. Our algorithm is localized (does not require any global topological knowledge), simple, and easy to implement. It can serve as a building block for implementing other non-trivial distributed coordination tasks in highly dynamic networks with churn.

Technical Overview. The main technical challenge that we have to overcome is designing and analyzing distributed algorithms with the presence of Byzantine nodes in networks where both nodes *and* edges can change by a large amount continuously in *each round*. The same challenge was present in solving the Byzantine agreement problem in such networks which was addressed in [4]. However, this does not directly solve the *leader election* problem, since the value that (most of) the honest nodes agree may be a value that was generated by a Byzantine node; using the agreement algorithm in a straightforward way does not give any guarantee that an honest node will be elected as leader. Hence, a more involved approach is needed for Byzantine leader election. We outline key ingredients of our approach here (Sections 2.1 and 2.2 give a more detailed overview).

While Byzantine agreement itself does not directly help, it can be used to generate an almost-everywhere public coin, i.e., an almost fair public coin that is known to most honest nodes. This is the first key ingredient. To the best of our knowledge, we present the first solution to such an almost everywhere (AE) common coin in a highly dynamic network.

Our protocol requires nodes to independently generate "lottery tickets" which are bit strings of certain length. Essentially, a node that has the winning lottery ticket becomes part of the small set of finalists from which a leader will be chosen eventually. However, there is a problem in naively implementing this approach. The Byzantine nodes, who know the current network state *including* random choices of other nodes, can change location and might lie about their lottery ticket number, thus claiming to be the winner. To overcome this, we implement

[6] In this paper, with high probability refers to a probability of $\geqslant 1 - n^{-\Omega(1)}$.

a *verification* mechanism that allows the honest nodes to check whether the Byzantine nodes are lying. This mechanism is as follows. Once a node generates its lottery ticket it "stores" it in about $\sqrt{n}$ (randomly chosen) nodes (exceeding the number of Byzantine nodes by an n^{ε} factor). To verify whether a node is indeed the owner of the lottery ticket that it claims, honest nodes will check with these $\sqrt{n}$ nodes. This prevents a Byzantine node from falsely claiming a lottery ticket that it did not generate in the first place. We show how such a storage and verification mechanism can be implemented efficiently despite the presence of Byzantine nodes in a dynamic network. The last ingredient of the protocol is an efficient and fault-tolerant mechanism to disseminate the identity of the leader to almost all the honest nodes.

We use random walks as the main tool for communication in our protocol. Previously, flooding techniques proved useful in solving the agreement problem under high churn but without Byzantine nodes [6]. In the presence of Byzantine nodes, however, flooding is less useful as it enables Byzantine nodes to disseminate lots of (corrupting) information along with those sent by honest nodes. On the other hand, if honest nodes use random walks (which are lightweight and local) in their information spreading protocol, the Byzantine nodes are no longer able to congest the network without bound. This proves crucial for getting a scalable protocol that uses only polylogarithmic message sizes per round and finishes in polylogarithmic rounds. We use a key technical result on random walks in a dynamic network with Byzantine nodes and adversarial churn called the "*dynamic random sampling* theorem" (cf. Theorem 2) that shows mixing properties of random walks (despite Byzantine nodes and large adversarial churn) and enables us to communicate efficiently among honest nodes.

Our protocol can tolerate up to $O(n^{\frac{1}{2}-\varepsilon})$ Byzantine nodes (for a small constant $\varepsilon > 0$) and up to $O(\sqrt{n}/\operatorname{polylog}(n))$ churn. This is essentially the best that our protocol can handle for two reasons. Our storage and verification mechanism needs to store each lottery ticket in about $\sqrt{n}$ nodes, and to be scalable in terms of the number of messages generated, it can handle only about $\sqrt{n}$ nodes (each such node generates a ticket, thus overall there will be about $n \operatorname{polylog}(n)$ messages — anything significantly more than this will cause much more congestion). The second reason is that for solving Byzantine agreement, we use a majority rule to progress towards agreement [12, 4]. This majority rule algorithm works as long as the number of Byzantine nodes are bounded by $O(\sqrt{n})$.

Related Work. There has been a lot of recent work on distributed agreement, byzantine agreement, and fault-tolerant protocols in dynamic networks. We refer to [17, 6, 4, 3] and the references therein for details on these works. Here we restrict ourselves to those works that are most closely related.

There has been significant work in designing peer-to-peer networks that are provably robust to a large number of Byzantine faults [15, 18, 27, 30]. These focus only on robustly enabling storage and retrieval of data items. The problem of achieving almost-everywhere agreement among nodes in P2P networks (modeled as an expander graph) is considered by King et al. in [23] in the context of the leader election problem; essentially, [23] is a sparse (expander) network implementation of the full information protocol of [22]. More specifically, [23]

assumes that the adversary corrupts a constant fraction $b < 1/3$ of the processes that are under its control throughout the run of the algorithm. The protocol of [23] guarantees that with constant probability an uncorrupted leader will be elected and that a $1-O(\frac{1}{\log n})$ fraction of the uncorrupted processes know this leader. We note that the algorithm of [23] does not work for dynamic networks, in particular, when just the edges are rewired from round to round (while still preserving expander topology). In another recent work [21], the authors use a spectral technique to "blacklist" malicious nodes leading to faster and more efficient Byzantine agreement in static networks. The idea of blacklisting, unfortunately, won't work in our dynamic network model, since the adversary can change the identities of Byzantine nodes by churning out old ones and introducing new ones (cf. Section 2).

The work of [6] addresses the agreement problem in a dynamic P2P network under an adversarial churn model where the churn rates can be very large, up to linear in the number of nodes in the network. (It also crucially makes use of expander graphs.) This introduced the dynamic model with churn that is used subsequently in other papers ([3, 4]), including this paper. However, the algorithms and techniques of [6] will not work under the presence of Byzantine nodes; even one malicious node can foil their algorithms. The work of [3] presented storage and search algorithms for a highly dynamic network that can have churn rate up to $n/\operatorname{polylog} n$. However, these algorithms do not work in the presence of Byzantine nodes. The random walk approach to dynamic sampling was introduced in this paper and subsequently extended to Byzantine nodes in [4]. [17] presents a solution for maintaining a clustering of the network where each cluster contains more than two thirds honest nodes with high probability in a setting where the size of the network can vary polynomially over time.

The work of [4] presents Byzantine almost-everywhere agreement algorithms that can tolerate the same amount of churn as the present paper, i.e., $\sqrt{n}/\operatorname{polylog} n$, but it works even under adaptive churn. For this algorithm, only the "rewiring" adversary, which controls the edges between the adaptively churned nodes, needs to be oblivious — this is needed for the random walk approach to work. We use this algorithm as a key building block for implementing our almost-everywhere common coin. [4] also presented an almost tight (up to $\operatorname{polylog} n$ factor) lower bound of $\Omega(\sqrt{n}/\operatorname{polylog} n)$ for the amount of churn that can be tolerated if one requires polylogarithmic round algorithms. This lower bound crucially makes use of the adaptive nature of the churn adversary. It is not clear if the same lower bound holds for the oblivious adversary as considered in this paper.

Expander graphs and spectral properties have been applied extensively to improve the network design and fault-tolerance in distributed computing (cf. [31, 14, 9]). The work of [26] provides a distributed algorithm for maintaining an expander in the presence of churn with high probability by using Hamiltonian cycles.

In recent years, adversarial models for dynamic networks have been studied extensively by [7, 11, 28, 25] and others; see the recent survey of [10] and the references therein. Unlike many early works on dynamic networks (e.g., [1, 13, 16, 2, 8]) these recent works do not assume that the network will eventually stabilize

and stop changing. On the other hand, we would prefer distributed algorithms to work correctly even if the network is *changing continuously over time* (as assumed in our paper). The works of [25, 7, 11] study a model in which the communication graph can change completely from one round to another, with the only constraint being that the network is *connected at each round* ([25] and [11] also consider a stronger model where the constraint is that the network should be an expander or should have some specific expansion in each round). The model has also been applied to agreement problems in dynamic networks; various versions of coordinated consensus (where all nodes must agree) have been considered in [25]. We note that the model of [24] allows only edge changes from round to round while the nodes remain fixed. The model considered here is more general than the model of [24], as it captures dynamic settings where edges change *and* nodes are subjected to churn. It is impossible to solve Byzantine agreement when only assuming the (oblivious) adversary of [24] that keeps the graph connected in each round; for example, a Byzantine node placed at a bottleneck point of the network can forever prevent any reasonable information flow between the two separated parts of the network. For the case where the "edge adversary" of [24] adheres to our expansion and regularity assumption, we can apply our results to this model as well.

1.1 Computing Model and Problem Definition

We consider a synchronous dynamic network with Byzantine nodes represented by a graph with a dynamically changing topology (both nodes and edges change) whose nodes execute a distributed algorithm and whose edges represent connectivity in the network. The computation is structured into synchronous rounds, i.e., we assume that nodes run at the same processing speed (and have access to a synchronized clock) and any message that is sent by some node u to its neighbors in some round $r \geqslant 1$ will be received by the end of r. The dynamic network is represented by a sequence of graphs $\mathcal{G} = (G^0, G^1, \ldots)$ where each $G^r = (V^r, E^r)$. Nodes might be subjected to churn, which means that in each round, up to $C(n)$ nodes ($C(n) \in O(\sqrt{n}/\log^k n)$) can be replaced by new nodes; the constant k will be fixed in the analysis. We require that, for all $r \geqslant 0$, $|V^r \setminus V^{r+1}| = |V^{r+1} \setminus V^r| \leqslant C(n)$. Furthermore, we allow the edges to change from round to round, but we assume that each G^r is a d-regular expander graph with constant spectral gap. The churn and the evolution of the edges are under the control of an *oblivious adversary* who has to choose the entire sequence of $(G^0, G^1, \ldots)$ in advance.

Up to $B(n) \in O(n^{\alpha/2})$ nodes can be *Byzantine* and deviate arbitrarily from the given protocol, where $\alpha > 0$ is a constant adhering to Equation(1) on Page 8. We say that a node u is *honest* if u is not a Byzantine node and use V_{corr} to denote the set of honest nodes in the network. Byzantine nodes are "adaptive", in the sense that they know the entire states of all nodes at the beginning of every round and thus can take the current state of the computation into account when determining their next action. This setting is commonly referred to as the *full information model*. We consider the usual assumption that Byzantine nodes cannot fake their identity, i.e., if a Byzantine node w sends a message to nodes u and v, then both u and v can identify w as the same sender of the message.

Note that this does not stop Byzantine nodes from forwarding fake messages on behalf of other nodes as we do not assume any authentication service. We assume, without loss of generality, that the adversary only subjects honest nodes to churn, i.e., Byzantine nodes remain in the network permanently. (The analysis of our algorithms can be extended easily to the case where Byzantine nodes are subject to churn as well).

We assume that if a node u enters the network at some later round r, then u knows the number of rounds that have passed since the start of the computation. Any information about the network at large is only learned through the messages that node u receives and u has no a priori knowledge about who its neighbors will be in the future.

We now describe the sequence of events that occur in each round $r \geqslant 1$. Firstly, we modify the network $G^{r-1} = (V^{r-1}, E^{r-1})$ by subjecting up to $C(n)$ nodes to churn (yielding V^r) and then changing the edge connectivity; recall that these changes are predetermined by the oblivious adversary. At this point, we emphasize that Byzantine nodes are always adaptive in the sense that they can observe the current network state including all past random choices. After the adversary has made its moves, the algorithm operates on the graph $G^r = (V^r, E^r)$ in round r. Each honest node u becomes aware of its current neighbors in G^r, can perform local computation and is able to reliably exchange messages with its neighbors according to the edges in E^r.

As stated above, our algorithm tolerates $O(n^{\alpha/2})$ Byzantine nodes and churn per round. Let $c_1 > 2$ be any fixed constant. Our algorithm requires the following condition on α:

$$\alpha \leqslant -\log(1 - 1/c_1)/\log c_1 - 8\log\log n/\log n - 1/c_1^2 \tag{1}$$

We now present the formal definition of the Byzantine leader election problem. Note that since we assume a dynamic network which is a *sparse* expander in each round, we cannot hope to obtain an algorithm where every honest node eventually knows the leader; for example, the adversary could simply keep churning out all neighbors of a node u, effectively isolating u throughout the run. This motivates us to consider the following "almost everywhere" variant of leader election:

Byzantine Leader Election (BLE). Suppose that there are $B(n)$ Byzantine nodes in the network. We say that an *algorithm A solves Byzantine Leader Election in T rounds* if, in any run of A, there is exactly 1 node u_ℓ such that

(a) all honest nodes terminate in T rounds whp,

(b) all except $B(n) + o(n)$ honest nodes accept u_ℓ as the leader, and

(c) node u_ℓ is honest with probability $\geqslant 1 - \frac{B(n)}{n} - o(1)$.

2 The Byzantine Leader Election Algorithm

In this section we present an algorithm for electing a leader in the presence of $O(n^{\alpha/2}/\operatorname{polylog}(n))$ Byzantine nodes and churn. Before presenting the details of our algorithm, we first discuss why more straightforward approaches do not work: At a first glance, it appears as if we might be able to simply use the Byzantine almost everywhere agreement (BAE) algorithm of [4] to elect a leader. For example, running bitwise BAE agreement on the node ids will inevitably

yield almost everywhere agreement on some specific node id. (After agreeing on the i-th bit v, we only consider nodes as candidates whose id has v as the i-th bit.) This, however, is poised to fail, since the adversary will simply choose the initial node ids in a way such that the BAE algorithm yields a decision on an id of a Byzantine node.

An immediate but insufficient improvement of the above approach is to initially instruct each honest node to generate a random id, and then run bitwise BAE agreement on these random ids to elect a leader. In this case, the oblivious adversary, who has to choose the churn and the initial nodes in advance, has no advantage in making an initial guess on the elected id. Byzantine nodes, on the other hand, have full knowledge of the current network state including past random choices in the full-information model. Thus, a Byzantine node u that announces an initially chosen value id_u, can adaptively lie about the actual value of id_u as soon as the outcome of the agreement algorithm becomes apparent, and subsequently claim leadership. Of course, if the network topology was static, the neighbors of u will notice that u has changed its initial value and could simply inform the remaining network to blacklist u as being malicious. In our model, however, the adversary has the power to rewire the topology over time and to subject nodes to churn, possibly causing all initial neighbors of u to be several hops away from u (or even churned out) during later rounds. This makes it difficult for an honest node v to conclude whether u has deviated from its initial choice, if u and v were not neighbors initially. In fact, any information that v has learned about u while not being a neighbor of u was learned indirectly via other nodes. As we neither assume an authentication service nor make any assumptions on how Byzantine nodes are distributed in the network, an easy indistinguishibility argument shows that v has no way of knowing if the learned information was injected by other Byzantine nodes.

2.1 Preliminaries and Technical Tools

Random Walk Implementation. To ensure lightweight communication costs, our algorithm relies on random walks as a means of communication. We now describe a simple token-passing implementation of random walks in our model (cf. [3, 4]): When an honest node u initiates a random walk, it generates a token with its id, a counter initialized to the length of the walk, and possibly attaches some piece of information of $O(\log n)$ size. This token is then forwarded to a (current) neighbor of u chosen uniformly at random, which in turn forwards the token and so forth. The counter is decreased by 1 each time the token is forwarded, until it reaches 0, which marks the final destination of this walk. Since Byzantine nodes can deviate arbitrarily from this protocol, honest nodes only forward tokens that are *legit*, which means that they adhere to above described data format. Our algorithm requires nodes to initiate $h \log n$ random walks simultaneously, for a sufficiently large constant h. During the run of the algorithm, an honest node u might receive a large number of tokens (possibly generated by Byzantine nodes). More precisely, the random walks that arrive at a node u are placed in a FIFO buffer according to the order of their arrival. To prevent Byzantine nodes from congesting the entire network with fake tokens, node u forwards up to $h \log n$ of

the tokens from its buffer in each step. This ensures (whp) passage of random walks that matter to us.

Our algorithm employs a technical result that shows almost uniform mixing for most random walks in our dynamic network. Its proof relies on a combination of several technical results and is related to Theorem 1 of [4]; we defer the details to the full paper and will focus on the new aspects of our leader election algorithm here. Intuitively speaking, Theorem 2 says that there is a large set Core of honest nodes such that, after walking for $\Theta(\log n)$ steps, tokens originating from these nodes have probability of $\approx 1/n$ to be at any node in Core. It is important to keep in mind that, since the size of Core is only guaranteed to be $\geqslant n - o(n)$, there is a nonzero probability for such a token to end up at nodes that are not in Core; for example, by being forwarded to a Byzantine node.

Theorem 2 (Dynamic Random Sampling). *Let $T = \Theta(\log^2 n)$ and consider a dynamic n-node expander network $\mathcal{G}$ under an oblivious adversary, and suppose that at most $O(n^{\alpha/2})$ nodes are Byzantine and at most $O(\sqrt{n}/\log^k n)$ nodes are subjected to churn in each round, where k is a sufficiently large constant and for any fixed constant $\alpha < 1$. Then, there exists a set of honest nodes Core of size $\geqslant n - O(\sqrt{n}/\log^{k-6} n)$ such that, in every time interval $[iT + 1, (i + 1)T]$ for $0 \leqslant i \leqslant \Theta(\log n)$ the following hold:*

1. *A random walk token originating from a node in Core has probability in $[\frac{1}{n} - \frac{1}{n^3}, \frac{1}{n} + \frac{1}{n^3}]$ to terminate at any particular node in Core.*
2. *At most $O(\sqrt{n}/\log^{k-8} n)$ nodes in Core receive tokens that did not originate in Core, and $\geqslant n - O(\sqrt{n}/\log^{k-9} n)$ nodes in Core only receive tokens that took all their steps among nodes in Core.*

Byzantine Almost-Everywhere Agreement. The following BAE agreement algorithm is given in [4]: Each honest node initially starts with an input bit (either 0 or 1) and instances of the random walk implementation by generating tokens that contain its input bit. Once such an instance is complete (after $\Theta(\log^2 n)$ rounds), each honest node tries to update its current input value with the majority value of the triple consisting of its input value and 2 of its received tokens. In particular, it follows from the analysis in [4] that $\Theta(\log n)$ repetitions suffice to converge to almost-everywhere agreement among all except $O(\sqrt{n}/\log^{k-6} n)$ nodes with high probability.

The following result lower bounds the number of nodes that agree in all instances when we run $\Theta(\log n)$ instances of this BAE agreement algorithm in parallel. (This is what we do when flipping the common coin in Phase 3 of our algorithm for choosing the winning lottery ticket.)

Corollary 1 (Parallel BAE agreement, follows from [4]). *Let $T = \Theta(\log^3 n)$ and suppose that at most $O(n^{\alpha/2})$ nodes are Byzantine, while up to $O(\sqrt{n}/\log^k n)$ nodes are subjected to churn in any round, for any constant $\alpha < 1$. Suppose that the honest nodes execute $\ell \leqslant \Theta(\log n)$ parallel instances of the BAE algorithm of [4]. Then, with high probability, there is a set $\mathsf{Agr} \subseteq \bigcap_{0 \leqslant r \leqslant T} V^r$ of honest nodes such that in each BAE agreement instance i ($1 \leqslant i \leqslant \ell$), all nodes in Agr decide on a common bit b_i within $T = \Theta(\log^3 n)$ rounds, and $|\mathsf{Agr}| = n - O(\sqrt{n}/\log^{k-7} n)$.*

Good and Bad Nodes. For convenience, we define $\mathsf{Bad}^r = V^r \setminus (\mathsf{Agr} \cup \mathsf{Core})$; that is, Bad^r is of size $O(\sqrt{n}/\log^{k-7} n)$, contains all Byzantine nodes, and all honest nodes that are in the network in round r and that are either not part of our Core set given by Theorem 2 or decided wrongly in at least one of the parallel BAE agreement algorithm instances. We also define the set $\mathsf{Good} = \mathsf{Agr} \cap \mathsf{Core}$.

2.2 A Byzantine Leader Election Algorithm

We now describe the details of our leader election algorithm and provide some intuition for its correctness.

Phase 1. Determining Candidates: To keep the overall message complexity per node polylogarithmic, we first subsample a set of candidates Cand, by instructing each node to randomly choose to become a candidate with probability $8 \log n/\sqrt{n}$. Our algorithm heavily depends on the sampling capabilities provided by Theorem 2. Recall that the churn and the changes of the communication links are chosen obliviously (cf. Section 1.1), while Byzantine nodes can adapt their behavior by taking into account the current network state. Intuitively speaking, the following lemma shows that the Byzantine nodes have no influence over which honest nodes end up in the set Core, as the Core set is solely determined by the churn and the topology changes, both of which are chosen in advance by the *oblivious* adversary (cf. Section 1.1):

Lemma 1 (Independence of Core). *The membership of nodes in the set Core (as defined in Theorem 2) is independent of the behaviour of the Byzantine nodes.*

Observing that each node chooses to become a candidate uniformly at random, it follows by a simple Chernoff bound that $|\mathsf{Cand}| \geqslant 4\sqrt{n} \log n$ whp. According to Lemma 1, the number of candidates that are in Core cannot be biased by the Byzantine nodes, but depend only on the churn and the topology changes, which are chosen in advance by an oblivious adversary. This motivates us to restrict ourselves to *core candidates* defined as $\mathsf{CCand} = \mathsf{Core} \cap \mathsf{Cand} \cap \mathsf{Agr}$, which are the candidates that agree in all instances of the BAE agreement algorithm (cf. Corollary 1) and are part of the Core set. From Corollary 1, it follows that $|\mathsf{Agr}| \geqslant n - o(\sqrt{n})$ and from Theorem 2 we know that $|\mathsf{Core}| \geqslant n - O(\sqrt{n}/\log^{k-6} n) \geqslant n - o(\sqrt{n})$. Therefore, the independence of Core from the behaviour of Byzantine nodes implies the following:

Corollary 2 (Number of agreeing core candidates). *With high probability, we have $|\mathsf{CCand}| \geqslant 2\sqrt{n} \log n$.*

Phase 2. Obtaining and storing lottery tickets: In this phase, we first instruct the candidate nodes to participate in the "leader lottery" by generating tickets. To this end, each candidate generates a lottery ticket represented as a private random bit string of length $\lceil \frac{\log n}{2 \log c_1} \rceil$, where $c_1 > 0$ is a constant depending on the bias of the "almost everywhere common coin" introduced in Phase 3. Note that all Byzantine nodes can pretend to be candidates and can collude to generate lottery tickets that maximize their chances. Next, we implement a storage mechanism to ensure that this information persists in the network despite the high churn rate and the dynamic topology changes.

Recall that we allow nodes to attach additional information onto their random walk tokens that they generate. Therefore, when referring to some information I communicated by a node v, we mean the additional information (of size $O(\log n)$) that v has piggybacked onto a random walk token message, as described in the random walk implementation (cf. Section 2.1). We say that *node u has stored information I in the network*, if u has generated I and there exist at least $\Theta(\sqrt{n}\log n)$ honest nodes that are witnesses regarding I. Keep in mind that Byzantine nodes are omniscient regarding the current network state, enabling them to claim to be witnesses for some arbitrary (possibly fake) information.

Since we assume a sparse network with a dynamically changing topology and only allow messages of polylogarithmic size, we cannot leverage techniques commonly used in static networks; in particular, we cannot bind Byzantine nodes to their initial choice by broadcasting this information to all nodes or requiring neighbors to keep track of each other's choices. Instead, we invoke a storage mechanism, which allows us to keep track of the initial choice of Byzantine nodes.

In more detail, we initiate the following branching process: When an (honest) candidate u invokes STORE(z), for some ticket z, it generates a random walk of sufficient length and piggybacks z onto the random walk token message. Suppose that the walk has reached only honest nodes and terminated at some honest node v. Node v in turn starts $\Theta(\log n)$ new random walks, each of which contains z. Each of these walks that reaches only honest nodes will in turn spawn $\Theta(\log n)$ new random walks and so forth; we repeat this branching process $\Theta(\log n)$ times. We can think of the branching process as creating a tree having $\Omega(\sqrt{n}\log n)$ leafs. Every honest node that corresponds to a leaf of this tree, locally stores z and becomes a *witness* of ticket z of node u. In the following, we say that *u plays ticket z* if z has been stored successfully.

Lemma 2. *Suppose that all candidates execute Procedure* STORE(I) *in parallel. Then, with high probability, each of the core candidate (i.e. set* $\mathsf{CCand} \subset \mathsf{Good}$*) is able to play its tickets.*

To ensure that nodes in Bad (which includes all Byzantine nodes) have a small chance to guess the winning ticket, we upper bound the number of distinct tickets that Byzantine nodes can play:

Lemma 3. *Let $\mathcal{I}$ be the set of distinct tickets generated by nodes in* Bad *such that each $I \in \mathcal{I}$ is stored in the network, i.e., I has $\Omega(\sqrt{n}\log n)$ (fake or honest) witnesses. Then $|\mathcal{I}| \in O(n^{\alpha/2}\log^4 n)$.*

Phase 3. Running the lottery to determine the winning ticket: While we cannot directly use Byzantine almost everywhere (BAE) agreement to obtain an honest leader with good probability, we will use the time-tested method of employing such an BAE agreement algorithm as a subroutine to obtain an almost-everywhere common coin (cf. Definition 1 below), which is one of the tools used by our algorithm. The goal of this phase is to determine the *finalists*, i.e., the nodes who generated the winning lottery ticket. To this end, we generate the winning ticket by flipping an almost-everywhere common coin:

Definition 1 (Almost Everywhere (AE) Common Coin). *Consider an algorithm P where every honest node outputs a bit and let Comm_Q be the event that all nodes in a set Q output the same bit value b. If there exist a constant $c_1 \geqslant 2$ and a set Q of size $n - o(n)$ such that (A) $\mathbb{P}[\mathsf{Comm}_Q] \geqslant 1 - n^{-\Omega(1)}$, and (B) $\frac{1}{c_1} \leqslant \mathbb{P}[b = 0 \mid \mathsf{Comm}_Q] \leqslant 1 - \frac{1}{c_1}$, then we say that P* implements an almost everywhere common coin (AE common coin) on set Q *and we say that P has* bias *at most $1/2 - 1/c_1$.*

We will now show that the BAE agreement algorithm given by Corollary 1 can be modified to yield such an AE common coin.

Theorem 3 (AE Common Coin). *Consider a synchronous dynamic n-node expander network under the control of an oblivious adversary where up to $B(n) = O(n^{\alpha/2})$ nodes are Byzantine, and suppose that up to $C(n) = O(\sqrt{n}/\operatorname{polylog}(n))$ nodes are subjected to churn per round. There exists a polylogarithmic messages and time algorithm that implements an almost everywhere coin on a set of $n - O(B(n) + C(n))$ nodes with a bias bounded by a constant $c < 1/2$.*

The honest nodes jointly perform $\lceil \log n / 2 \log c_1 \rceil$ flips of this AE common coin to yield the winning ticket that will be known to almost all nodes.

Lemma 4. *We partition the set of stored tickets into the set $\mathsf{CoreTickets}$ of tickets generated by nodes in Good and the set $\mathsf{BadTickets}$, which contains the tickets played by (honest and Byzantine) nodes in Bad. Consider the winning lottery ticket s yielded by the $\lceil \frac{\log n}{2 \log c_1} \rceil$ invocations of the AE common coin algorithm. Then, it holds that (a) $\mathbb{P}[\forall x \in \mathsf{BadTickets}\colon x \neq s] \geqslant 1 - n^{-\Omega(1)}$. (b) $\mathbb{P}[\exists y \in \mathsf{CoreTickets}\colon y = s] \geqslant 1 - n^{-\Omega(1)}$.*

Recalling that the bits of the winning ticket comprises exactly the common decision values of the parallel BAE agreement instances, it follows that all nodes in set Agr know the winning ticket. Thus, each $u \in \mathsf{Agr}$ knows whether it is itself a winner (and thus becomes a *finalist*) or if it is among one of the $\Theta(\sqrt{n} \log n)$ witnesses of the finalist nodes. If so, u adds itself to the set of *propagators* P_v, for finalist v.

Phase 4. The final competition and leader election: In the final phase, one of the finalists must be chosen as the leader despite the fact that Byzantine nodes can behave like finalists and/or witnesses. In particular, we wish to reach a consensus on the finalist f with the smallest id.

We subdivide Phase 4 into $\frac{\log n}{2 \log \log n} + \Theta(1)$ sub-phases, each of $O(\log^2 n)$ rounds. Each honest node u samples $O(\log n)$ nodes per sub-phase via random walks. During this sampling process, u tries to discover the finalist f with the smallest id. Node u maintains a variable MIN-ID initialised to ∞. During phase 4, the honest nodes will only pass $O(\log n)$ random walk tokens per time step. At the start of each sub-phase, every honest node u initiates $\Theta(\log n)$ random walk tokens: if u is a witness for a ticket, then that ticket and the MIN-ID value are included in the token; the token is blank otherwise[7]. Each random walk must take

[7] The blank tokens cannot be discarded because they provide the congestion required to ensure that the number of tokens injected by Byzantine nodes are kept in check.

$\Theta(\log n)$ random walk steps in order to mix; this can be achieved in $O(\log^2 n)$ rounds (cf. Theorem 2). At the end of the sub-phase, each node looks at all the tokens that terminated on it and checks to see if v has an id smaller than its current MIN-ID and, if needed, updates MIN-ID with the smaller id. We now argue that at the end of $\frac{\log n}{2 \log \log n} + \Theta(1)$ sampling sub-phases, $n - o(n)$ nodes will have their MIN-ID set to f. This completes the proof of Theorem 1.

Lemma 5. *Let* Finalists *be the set of all candidates in* CCand *that played the winning ticket z and assume that z was stored among $\Omega(\sqrt{n} \log n)$ honest witnesses. Suppose that $f \in$ Finalists is the node with the smallest id in* Finalists*. Then, by the end of Phase 4, $n - o(n)$ nodes accept f as the leader with probability at least $1 - o(1)$.*

3 Conclusion

In this paper, we take a step towards designing secure, robust, and scalable algorithms for large-scale dynamic networks. We presented a scalable and lightweight distributed protocol for the fundamental Byzantine leader election in dynamic networks, tolerating near $O(\sqrt{n}/\operatorname{polylog}(n))$ Byzantine nodes and churn per round while using only polylogarithmic amount of messages per node. A key open problem is to show a lower bound that is essentially tight with respect to the amount of Byzantine nodes that can be tolerated, or show a leader election algorithm that can tolerate significantly more Byzantine nodes and churn. The latter might be possible, since we are dealing with an oblivious churn adversary (unlike the adaptive churn adversary of [4]).

References

1. Yehuda Afek, Baruch Awerbuch, and Eli Gafni. Applying static network protocols to dynamic networks. In *FOCS'87*, pages 358–370, 1987.
2. Yehuda Afek, Eli Gafni, and Adi Rosen. The slide mechanism with applications in dynamic networks. In *ACM PODC*, pages 35–46, 1992.
3. John Augustine, Anisur Rahaman Molla, Ehab Morsy, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Storage and search in dynamic peer-to-peer networks. In *SPAA*, pages 53–62, 2013.
4. John Augustine, Gopal Pandurangan, and Peter Robinson. Fast byzantine agreement in dynamic networks. In *PODC*, pages 74–83, 2013.
5. John Augustine, Gopal Pandurangan, Peter Robinson, Scott Roche, and Eli Upfal. Enabling efficient and robust distributed computation in highly dynamic networks. In *FOCS*, 2015. (To appear.).
6. John Augustine, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. In *SODA*, pages 551–569, 2012.
7. Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *ICALP*, pages 121–132, 2008.
8. Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Michael E. Saks. Adapting to asynchronous dynamic networks. In *STOC'92*, pages 557–570, 1992.
9. Amitabha Bagchi, Ankur Bhargava, Amitabh Chaudhary, David Eppstein, and Christian Scheideler. The effect of faults on network expansion. *Theory Comput. Syst.*, 39(6):903–928, 2006.

10. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *CoRR*, abs/1012.0009, 2010. Short version in ADHOC-NOW 2011.
11. Atish Das Sarma, Anisur Molla, and Gopal Pandurangan. Fast distributed computation in dynamic networks via random walks. In *DISC*, 2012.
12. Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. Stabilizing consensus with the power of two choices. In *SPAA*, pages 149–158, 2011.
13. Shlomi Dolev. *Self-stabilization*. MIT Press, Cambridge, MA, USA, 2000.
14. Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree. *SIAM J. Comput.*, 17(5):975–988, 1988.
15. Amos Fiat and Jared Saia. Censorship resistant peer-to-peer content addressable networks. In *SODA*, pages 94–103, 2002.
16. E. Gafni and B. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Trans. Comm.*, 29(1):1118, 1981.
17. Rachid Guerraoui, Florian Huc, and Anne-Marie Kermarrec. Highly dynamic distributed computing with byzantine failures. In *PODC*, pages 176–183, 2013.
18. Kirsten Hildrum and John Kubiatowicz. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *DISC*, volume 2848 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2003.
19. Bruce M. Kapron, David Kempe, Valerie King, Jared Saia, and Vishal Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. *ACM Transactions on Algorithms*, 6(4), 2010.
20. Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: scalable Byzantine agreement with an adaptive adversary. In *PODC*, pages 420–429, 2010.
21. Valerie King and Jared Saia. Faster agreement via a spectral method for detecting malicious behavior. In *SODA*, pages 785–800, 2014.
22. Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *SODA*, pages 990–999, 2006.
23. Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *FOCS*, pages 87–98, 2006.
24. Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *ACM STOC*, pages 513–522, 2010.
25. Fabian Kuhn, Rotem Oshman, and Yoram Moses. Coordinated consensus in dynamic networks. In *PODC*, pages 1–10, 2011.
26. C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2133 – 2143 vol.3, march-3 april 2003.
27. Moni Naor and Udi Wieder. A simple fault tolerant distributed hash table. In *IPTPS*, pages 88–97, 2003.
28. Regina O'Dell and Roger Wattenhofer. Information dissemination in highly dynamic graphs. In *DIALM-POMC*, pages 104–110, 2005.
29. Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Building low-diameter p2p networks. In *FOCS*, pages 492–499, 2001.
30. Christian Scheideler. How to spread adversarial nodes?: rotate! In *STOC*, pages 704–713, 2005.
31. Eli Upfal. Tolerating a linear number of faults in networks of bounded degree. *Inf. Comput.*, 115(2):312–320, 1994.